УД К 004.9

Ашихмін А. В.

РОЗРОБЛЕННЯ ПАТЕРНУ НАЛАШТУВАННЯ МУЛЬТИМЕДІЙНОЇ СИСТЕМИ

Задача налаштування системи є типовою в розробленні більшості мультимедійних проектів. Головна ідея налаштування полягає в керуванні системою в реальному або відкладеному часі шляхом змінення значень системних параметрів. З одного боку, збільшення кількості керованих параметрів системи підвищує її гнучкість, з іншого – ускладнює керування. Ключовим є той факт, що розроблення розвиненої підсистеми налаштування є досить складним та дорогим завданням, вирішення якого, тим не менш, завжди має схожі повторювані кроки. Отже, природним є питання автоматизації цього розроблення. Статтю присвячено опису процесу створення універсального патерну підсистеми налаштування. Патерн можна застосувати до будь-якого розроблення, що потребує керування в реальному часі або відкладеному режимі, але в першу чергу призначене для мультимедійних систем із оброблення графіки, аудіо- та відеосигналів. Патерн відповідає таким вимогам, актуальним у час стрімкого інформаційного розвитку: цілісність, прозорість, гнучкість, масштабованість, модульність, автоматизація. Найціннішою та перспективною перевагою використання патерну є механізм автоматичної генерації інтерфейсу користувача за набором вхідних системних параметрів.

Поняття налаштування

Налаштування є багатозначним поняттям, яке поєднує такі аспекти з точки зору процесу розроблення ПЗ: конфігурування, конфігурація, керування, ініціалізація, стан системи, підтримка профілів та ін. Налаштування системи є типовою задачею, що ставиться в переважній більшості застосувань. За визначенням, налаштуванням (settings) є параметри системи чи операції, що можуть бути обрані користувачем [1]. Процес налаштування може відбуватись у режимі реального часу, тоді він називається керуванням системою. Налаштування у відкладеному режимі прийнято називати конфігуруванням. Конфігурацією є набір значень параметрів, що система має під час ініціалізації [2]. Керування, як і конфігурування, полягає у встановленні та зміні значень окремих параметрів системи. Іншим аспектом налаштування є власне значення параметрів, які визначають поточний стан системи. Підвищення кількості керованих параметрів, з одного боку, надає більшу гнучкість, з іншого – призводить до ускладнення керуванням системою. Більше того, програмування такої системи налаштування є складною роботою, що потребує значної частки ресурсів від загального часу розроблення. Саме

тому природно випливає задача автоматизації процесу розроблення підсистеми налаштування та побудови зручного інтерфейсу користувача на її основі.

У ході аналізу ряду сучасних мультимедійних застосувань, таких як Cubase, Adobe Audition, Reaktor, Winamp, Wavelab, Nuendo, Audio Logic, AAMS тощо, сформовано перелік типових задач, що виникають під час розроблення підсистем налаштування. Вирішення цих задач зазвичай відбувається за типовою схемою, саме тому буде доцільним об'єднати їх в одному патерні – налаштування.

Отже, визначимо **основні вимоги до патерну налаштування:**

1. Ієрархічна організація параметрів, підтримка наслідування та агрегації, можливість повторення структури проекту.
2. Забезпечення локалізації (багатомовності).
3. Збереження стану системи в різноманітних фізичних середовищах (реєстр, INI-файл, пам'ять, база даних), профілях. Імпорт профілю за замовчуванням. Урахування версій продукту.
4. Підтримка множинності параметрів (кілька параметрів одного типу)
5. Підтримка багатопоточності.
6. Уніфікований унікальний доступ до кожного параметра налаштування.
7. Автоматична генерація графічного інтерфейсу.
8. Забезпечення налаштування та збереження обраних одиниць виміру.
9. Сумісність та підтримка архітектури Document / View або іншої.

Опис підходу до реалізації:

1. Підтримка ієрархії налаштувань, підтримка наслідування та агрегації

Стосовно логічної організації налаштувань доцільно використати деревоподібну структуру даних, приклад якої наведено на рис. 1. Кожен лист дерева, що зображено, є окремим параметром системи. Параметри об'єднано в групи параметрів, або складні групи, що об'єднують більш прості. Ступінь вкладеності груп є необмеженим. Групам відповідають вузли дерева. Корінь дерева – найбільш складна група, що містить усі параметри. Кореню відповідає власне об'єкт налаштування. Мета побудови деревоподібної ієрархії – організація налаштувань таким чином, щоб вони якнайкраще відповідали внутрішній логічній структурі параметрів застосування й забезпечення легкого та ефективного доступу до кожного з них.

Глобальний об'єкт налаштування є коренем дерева. Листи відповідають параметрам, а всі інші вузли дерева являють собою групи параметрів, що можуть бути вкладеними та обов'язково мають корінь та синів.

З одного боку, представлення параметрів у вигляді дерева є зручним тому, що можна точно відобразити їхню внутрішню структуру, з іншого – шляхом обходу дерева їх можна представити у вигляді лінійного списку, що складатиметься з параметрів і / або груп параметрів.

2. Локалізація (багатомовність)

Мета локалізації – забезпечити підтримку багатомовності застосування. У контексті розроблення патерну «Налаштування» ставиться за мету якомога повніше автоматизувати та спростити програмування підтримки мови.

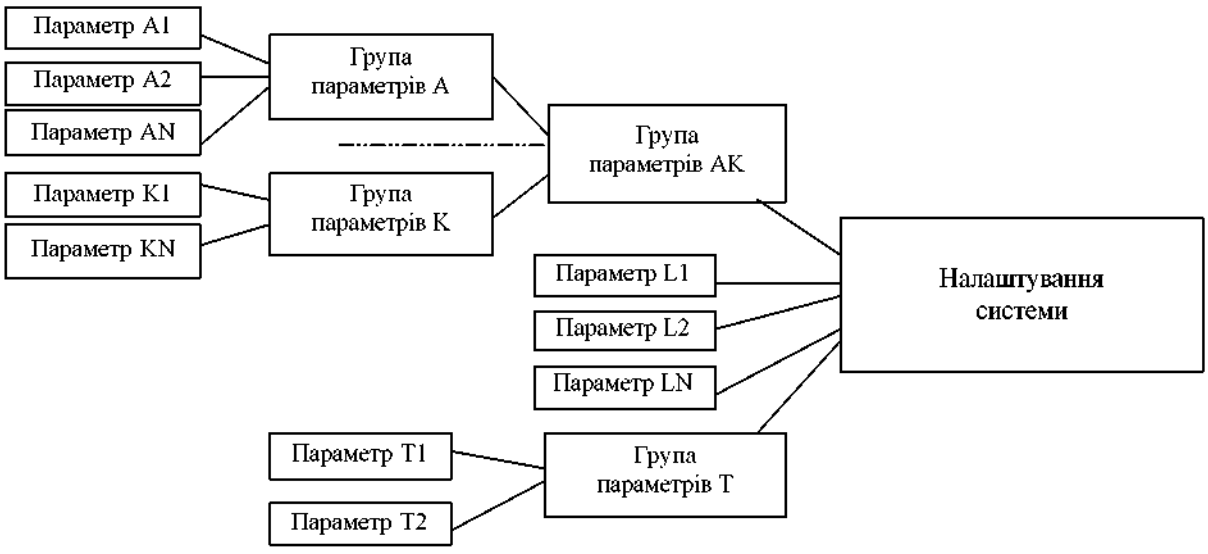

Рис. 1. Організація параметрів у вигляді дерева

Для локалізації використовують файли ресурсу, структуровані на зразок файлів .mc, що їх створює Microsoft message compiler [3], – кожний рядок представлено у форматі стандартного Windows-повідомлення.

У цьому файлі кожному параметру відповідає блок, де кожне представлення рядка параметра складається з таких атрибутів параметра:

1) IDV – унікальний ідентифікатор параметра або групи параметрів.

2) IDV_VALUE – символьне значення параметра.

3) LANGUAGE – мова.

4) Рядок опису, що має такий формат: <повний опис> "|"<короткий опис>"|"<абревіатура>.

Повний опис використовують для відображення повної інформації про параметр, що має бути зрозумілою недосвідченому користувачу. Має відображатися або в підказці (tooltip), або в лінійному переліку всіх параметрів.

Короткий опис використовується для відображення скороченої інформації про параметр, яка має бути зрозумілою користувачу, що вже приблизно обізнаний із внутрішньою структурою системи. Використовують у інтерфейсі з невеликою щільністю параметрів.

Абревіатура – найкоротша інформація про програму, бажано до п'яти символів. Використовують у варіанті інтерфейсу з великою щільністю параметрів, коли відбувається накопичення елементів.

3. Внутрішня таблиця опису параметрів

Для того, щоб забезпечити зв'язок між багатомовним ресурсом та структурами даних застосування, будується таблиця, в якій зберігатиметься вся мовно незалежна інформація про кожен параметр системи. Другорядною метою таблиці є забезпечення роботоспроможності системи у випадку, якщо виявилось, що файлу ресурсів не існує. Тобто можна включити текстовий опис мовою за замовчуванням у разі, якщо файл ресурсів відсутній.

Таблиця орієнтовно складається з таких полів: 1) ідентифікатор параметра; 2) ідентифікатор одиниці вимірювання; 3) ідентифікатор типу даних; 4) текстовий опис; 5) клас-елемент керування за замовчуванням; 6) значення за замовчуванням; 7) максимальне значення; 8) мінімальне значення; 9) інкремент, або крок; 10) шкала X (для підтримки графіків); 11) шкала Y; 12) тип графічного елемента керування; 13) вказівник на наступну структуру (або рядок таблиці) – для обходу таблиці доцільно використати шаблон «Ітератор» [6]. Вказівник на наступний рядок забезпечує механізм обходу всіх рядків таблиці. Оскільки сама таблиця є глобальним об'єктом, що не матиме копій, слід імплементувати її за шаблоном Singleton [5].

4. Збереження в різноманітних фізичних середовищах, профілі

Однією з найважливіших функцій будь-якої системи налаштування є можливість збереження параметрів системи в певному сховищі даних. Прикладами такого сховища можуть бути: реєстр, Reg-файл, Ini-файл, файл у пам'яті або на диску, база даних, файл у мережі або на ftp-сервері.

Отже, сховище може мати будь-який спосіб фізичного представлення. Саме тому варто організувати прив'язку до абстрактного сховища даних, що не містить інформації про конкретну реалізацію представлення даних. Для цього доцільно використати патерн «Абстрактна фабрика» [4]. Абстракне сховище слід наділити такими функціями: 1) завантаження; 2) збереження параметрів; 3) отримання елементів сховища. Для реалізації останнього пункту доцільно наділити сховище поведінкою за шаблоном «Ітератор» [6]: 3а) отримання першого елемента-параметра сховища; 3б) отримання наступного елемента.

5. Імпорт профілю за замовчуванням

Поняття «профіль за замовчуванням» має два контексти:

1) Початковий стан системи, коли вона запускається вперше. Цей профіль визначається значенням за замовчуванням внутрішньої таблиці опису параметрів. Назвемо його «базовий профіль».

2) Стан системи за замовчуванням, що обрано адміністратором системи. Це стан системи, що збережено в особливий профіль, який завантажуватиметься в кожному наступному запуску системи.

6. Врахування версій

Ведення версій у побудові підсистеми налаштувань важливо з таких міркувань. Якщо внутрішня структура проекту суттєво змінилась, підняття зі сховища налаштувань, які були збережені попередніми версіями ПЗ, може призвести до неочікуваних результатів, навіть до збоїв системи [7]. Підтримка версій може бути забезпечена за рахунок унікальності ідентифікаторів (IDV_). Для кожної нової версії продукту можна створювати новий кореневий елемент дерева, таким чином навіть за однакової структури ієрархії можна відрізнити налаштування, виконані

однією версією програми, від інших. Інший варіант реалізації врахування версії – записати версію налаштування безпосередньо, як значення кореневого елемента налаштування.

7. Підтримка множинності налаштувань

Під цим розуміється можливість розміщувати в ієрархії кілька об'єктів одного типу на одному рівні ієрархії. Це необхідно в разі, коли маємо справу з великою кількістю однотипних параметрів – тобто вони мають однакову одиницю виміру, однакові межі змінення, крок та значення за замовчуванням. Тоді, використовуючи підтримку множинності, можна позбавитися зберігання надлишкової інформації про об'єкт. Типовим прикладом таких налаштувань є положення регуляторів багатополосного еквалайзера.

8. Підтримка багатопоточності

Зазвичай модифікація параметрів відбувається в одному потоці – віконному, а зчитування – у віконному та робочому. Під робочим потоком мається на увазі потік, у якому відбувається оброблення даних у реальному часі, відповідно до поточних налаштувань, які можуть змінюватись користувачем у реальному часі, тобто одночасно з обробленням. Саме тому є нагальна потреба в забезпеченні механізму міжпотокового блокування. Коли змінюємо певний параметр, має відбутися коректне врахування в потоці-обробнику. З точки зору реалізації та відлагодження, найпростішим способом блокування параметра є блокування всієї ієрархії налаштування, оскільки необхідно відстежувати лише один об'єкт синхронізації – той, що належить кореню системи. Дотримання такої простоти блокування є цілком виправданим, оскільки налаштування зазвичай змінюються користувачем поступово, а не одночасно, і з відносно невеликою швидкістю, у порівнянні зі швидкістю роботи об'єкта синхронізації.

9. Уніфікований унікальний доступ кожного параметра налаштувань

У будь-який момент часу необхідно мати можливість доступу до будь-якого параметра системи, якщо системою не передбачено навпаки.

Доступ до параметра відбувається за його ідентифікатором. Тому слід реалізувати генерацію унікального ідентифікатора параметра, що складається з повного шляху від кореня та його імені.

Розглянемо рис. 2.

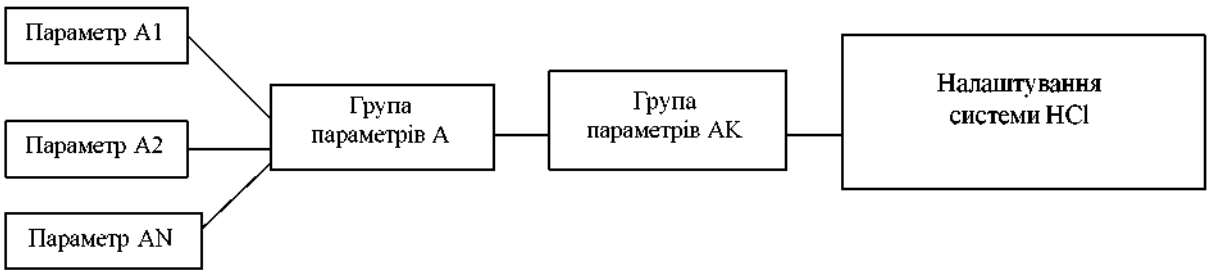

Рис. 2. Логічне розташування параметрів, за якими будується ідентифікатор

У разі збереження налаштувань в ini-файл ідентифікатори трьох параметрів системи матимуть вигляд:

[HC1.AK.A.A1], **[HC1.AK.A.A2]** та **[HC1.AK.A.A3]**; розділовим знаком є крапка.

У збереженні налаштувань у реєстр ідентифікаторами параметрів системи будуть шляхи до реєстру, тобто:

[HKCU\Sofware\CompanyName\SETTINGS\HC1\AK\A\A1],
[HKCU\Sofware\CompanyName\SETTINGS\HC1\AK\A\A2],
[HKCU\Sofware\CompanyName\SETTINGS\HC1\AK\A\A3].

Зважаючи на те, що параметри ідентифікуємо числами, повний ідентифікатор параметра A1 матиме вигляд:

[42309403.42309445.42309563.42309975], у збереженні в ini-файл та **[HKCU\Sofware\CompanyName\SETTINGS\ HC1\42309403\ 42309445\42309563\42309975]**.

За аналогічним принципом слід генерувати ідентифікатори і для інших типів сховищ.

Перевагою такої форми збереження є нечитабельність, що ускладнює зміни параметрів вручну та не дає можливості змінювати систему іншим чином, окрім конфігураційної утиліти. Здебільшого такий спосіб запису захистить від небажаних помилок.

10. Автоматична генерація графічного інтерфейсу

Найбільш привабливою можливістю підсистеми налаштувань є можливість автоматичної

генерації графічного інтерфейсу користувача. Мета цієї властивості – забезпечити інтерфейс за замовчуванням, що має свої переваги. Серед них – суттєве скорочення часу розроблення, оскільки зазвичай програмування інтерфейсу займає близько третини загального часу.

Одна з чудових переваг автоматично генерованого інтерфейсу – моніторинг та модифікація параметрів або інших змінних підсистеми під час розроблення системи, її налагодження та тестування як простих, так і складних модулів. Код, необхідний для підтримки такої системи моніторингу, буде невеликим: достатньо лише змінити тип змінних та викликати функцію ініціалізації діалогу. У будь-який момент розроблення його можна буде відключити.

Власне задача генерації інтерфейсу – до імплементації алгоритму автоматичного розташування графічних елементів керування, оскільки патерн передбачає зручний доступ до інформації про логічну структуру параметрів керування, мовнонезалежні текстові рядки, а також можливість збереження та відновлення станів налаштування, що дає змогу імплементувати механізми Undo/Redo, а також імпорту та експорту профілів.

11. Підтримка відкладеного режиму та режиму реального часу

Система має працювати у двох режимах: реальному та відкладеному. Реальний режим часу: такі зміни параметрів, на які система має реагувати моментально [8].

Застосування: створення, або генерація інтерактивного інтерфейсу з користувачем. Внутрішнє керування параметрами одного модуля іншим, управляючим.

У відкладеному режимі часу відбувається інсталяція, конфігурування та адміністрування системи. Відповідно цей режим застосовуватиметься в інсталяторах, конфігураційних вікнах, адміністративних панелях.

12. Забезпечення налаштування та збереження обраних одиниць виміру

Кожен параметр – не просто абстрактне числове значення, воно має свій контекст – одиницю виміру. Більшість одиниць є стандартними, наприклад, децибели, байти, години, секунди, тому є сенс автоматизувати цю частину системи. Важлива функція – вибір одиниці представлення у графічному інтерфейсі.

Підсистему вибору одиниць виміру можна будувати аналогічно у вигляді таблиці одиниць виміру та методів доступу до них, де кожна одиниця має свій ідентифікатор, аналогічно до таблиці параметрів. До цієї таблиці можна звертатися з будь-якого місця програми за допомогою відповідного API.

13. Сумісність та підтримка архітектури Document / View

Document/View є типовою архітектурою сучасного Windows-застосування. Тому її підтримка є цілком виправданою, якщо зберігається умова універсальності – можливості використання у довільній архітектурі.

Тут використовують патерн «Adapter». Інтерфейс має бути достатньо прозорим та інтуїтивним, щоб погляду досвідченого системного програміста було достатньо, щоб зрозуміти, яким чином він має бути вбудований, які функції класів CDocument та CView слід перевизначати, які повідомлення від віконного інтерфейсу слід обробляти і відповідно які елементи інтерфейсу мають бути в системі.

Висновки

Описано патерн, за яким слід розроблювати та використовувати систему налаштування. Створено список вимог до функціональності. Наведені функції є необхідними для більшості мультимедійних систем, що працюють у реальному або відкладеному часі. Система налаштування, що побудована за цими вимогами, може бути швидко вбудована в розроблений наполовину проект та використовуватись як інструмент відлагодження. Але більш доцільним застосуванням буде використання такої підсистеми як початкової точки проекту, оскільки це передбачає ретельний підхід до побудови архітектури системи. Чітке бачення способу розроблення та автоматизація побудови складових системи є вирішальним фактором для досягнення прозорості та прискорення процесу розроблення та мінімізації кількості допущених помилок під час проектування та імплементації системи [7].

Головна ідея патерну – уособлення зв'язку між функціями системи, які перелічено вище. Він містить більш прості відомі патерни: Abstract Factory, Singleton, Adapter, Façade, Iterator. Важливою рисою є поєднання найнижчого рівня системи – алгоритмічного, проміжних логічних рівнів та рівня програмного інтерфейсу користувача, або інсталятора. Дотримання патерну може стати суттєвим кроком у напрямку отримання вкрай важливих властивостей проекту, таких як цілісність, прозорість, гнучкість, масштабність, модульність, автоматизованість.

1. Glossary.– http://www.micro2000.co.uk/products/remotescope/glossary.htm
2. Glossary.– http://www.massmind.org/techref/glossary.htm
3. Microsoft Message Compiler.– http://msdn2.microsoft.com/en-us/library/aa385633.aspx
4. Abstract Factory Design Pattern.– http://www.exciton.cs.rice.edu/JAvaResources/DesignPatterns/FactoryPattern.htm
5. Singleton Design Pattern.– http://www.dofactory.com/Patterns/ PatternSingleton.aspx
6. Iterator Pattern.– http://en.wikipedia.org/wiki/Iterator_pattern
7. *Gamma E., Helm R., Johnson R., Vissides J.* Design Patterns: Elements of Reusable Object-Oriented Software.– New York: Addison-Wesley, 1994.
8. What is Real-Time Processing? – http://media.wiley.com/product_data/excerpt/90/08194178/0819417890-2.pdf

A. Ashyhmin

DESIGN PATTERN OF THE SETTINGS MULTIMEDIA SYSTEM

System settings design is a typical task in development of the most multimedia projects. The main idea of settings is system control in real time or suspended mode. It consists in setting and changing values of particular system parameters. On one hand, the more changeable parameters are present, the higher flexibility of a system is provided. On another hand, parameter increase leads to greater complexity of system control. More important thing is that development of a complex subsystem of settings is a difficult task which usually consumes an essential part of the whole project resources. However, there are many repetitive steps, similar for the most projects. Thus, the problem of automation of settings subsystem development appears. This work is dedicated to the description of creation of a universal design pattern for such a subsystem. The pattern may be applied to any system development where real-time or suspended mode control is necessary, but mainly it is intended for multimedia systems which deal with graphics, audio, and video signal processing. It allows reaching the properties highly important in the context of the current high-speed information technologies evolution: integrity, clarity, flexibility, scalability, modularity, automation. The most valuable and prospective advantage is the automatic generation of graphic user interface by a set of input system parameters.